Explorer une grille avec un minimum de robots amnésiques

Franck Petit, Anissa Lamani, Stéphane Devismes, Sébastien Tixeuil, Pascal Raymond

Explorer une grille avec un minimum de robots amnésiques †

Stéphane Devismes[1], Anissa Lamani[2], Franck Petit[3], Pascal Raymond[1] et Sébastien Tixeuil[3]

[1] *VERIMAG UMR 5104, Université Joseph Fourier, Grenoble* [2] *MIS, Université de Picardie Jules Verne, Amiens*
[3] *LIP6 UMR 7606, UPMC Sorbonne Universités, Paris*

Nous nous intéressons à l'exploration finie d'une grille $i \times j$ par une équipe de k robots autonomes, asynchrones et amnésiques — i et j représentent le nombre de lignes et de colonnes de la grille avec $1 \leq i \leq j$. Nous montrons que $k = 3$ robots sont nécessaires et suffisants pour résoudre le problème dans les modèles asynchrone et semi-synchrone, sauf si $j \leq 2$ ou si $i = j = 3$, auxquels cas k doit respectivement être égal à $i \times j$ ou au moins égal à 5.

Keywords: exploration, grille, robots, minimalité

1 Introduction

Nous considérons k *robots* (ou *agents*) autonomes se déplaçant dans une grille $i \times j$ non orientée telle que i et j représentent le nombre de lignes et de colonnes de la grille — par convention, nous supposons $1 \leq i \leq j$. Nous nous intéressons au problème de l'*exploration finie* d'une telle grille : à partir d'une configuration où les k robots sont arbitrairement placés sur des nœuds distincts, la grille doit être entièrement visitée par les robots, avant que ceux-ci ne s'arrêtent définitivement. Plus précisément, la terminaison doit être certaine et lors de cette terminaison, chaque nœud de la grille doit avoir été visité par au moins un des robots. Les robots que nous considérons sont dotés de capteurs visuels et d'actionneurs de mouvement. Leur programme consiste à exécuter infiniment souvent le cycle `OCD` constitué des trois phases suivantes : `Observation`, `Calcul` et `Déplacement`. Durant la phase d'observation, le robot prend connaissance de son environnement à l'aide de ses capteurs visuels. Ensuite, en fonction de l'environnement perçu, le robot décide de se déplacer sur l'un des nœuds adjacents ou de rester sur place. Les robots n'ayant pas accès à un dispositif partagé de synchronisation, les trois phases sont exécutées de manière complètement asynchrone par chacun des robots. Cependant, nous supposons qu'un robot ne peut pas être vu en transit entre deux nœuds adjacents ‡. Par ailleurs, rien ne permet de distinguer un robot d'un autre et tous exécutent le même programme. De plus, les robots sont *amnésiques*, c'est-à-dire que leur mémoire est volatile et s'efface complètement entre deux cycles `OCD`. Enfin, les robots sont dénués de tout moyen de communication direct. En revanche, lors de la phase d'observation, ils perçoivent la grille dans son intégralité et sont capables de déterminer pour chaque nœud s'il contient 0, 1 ou plusieurs robots. Néanmoins, ils ne disposent d'aucun dispositif qui leur permettrait de déterminer une quelconque orientation de la grille ou d'identifier les nœuds *a priori*. Ainsi, suite à une observation, il se peut que plusieurs arêtes adjacentes au nœud occupé par le robot lui semblent identiques, c'est-à-dire que le robot se situe alors sur un axe de symétrie de la configuration. Dans ce cas, si le robot décide de bouger, il peut traverser l'une ou l'autre des arêtes : nous considérons alors le pire cas où le choix de l'arête traversée est décidé par un adversaire.

L'exploration finie par k robots a été étudiée pour des anneaux de n nœuds [1, 2, 3]. Dans [1], les auteurs montrent que le problème peut être résolu de manière déterministe seulement si k et n sont premiers entre

†La version longue de cet article a été présentée lors de SSS'2012.

‡. Cette hypothèse est à la fois usuelle et non restrictive dans ce type de modèle. D'une part, elle simplifie les algorithmes. D'autre part, elle peut être levée facilement : le robot peut décider de ne pas bouger lorsqu'il observe que des robots ne sont pas sur des nœuds.

eux. Ils proposent d'ailleurs un algorithme pour ce cas, celui-ci nécessite en plus que k soit au moins égal à 17. Il est prouvé dans [2] que 4 robots sont nécessaires et suffisants pour résoudre l'exploration avec une probabilité 1 de terminaison dans le modèle semi-synchrone, sans l'hypothèse de primalité entre k et n. Dans le modèle semi-synchrone, à chaque instant t, un adversaire sélectionne un sous-ensemble non vide de robots, lesquels exécutent intégralement le cycle `OCD` entre t et $t+1$; de plus, l'adversaire doit sélectionner tout robot infiniment souvent. Enfin, un algorithme déterministe dans le modèle asynchrone utilisant un minimum de robots, *i.e.*, 5, est proposé dans [3]. Suivant le résultat de [1], cet algorithme ne considère que des anneaux de taille non divisible par 5.

Dans cet article, nous démontrons que seuls $k = 3$ robots sont nécessaires et suffisants pour résoudre l'exploration (déterministe) finie d'une grille $i \times j$ avec $1 \leq i \leq j$ dans les modèles asynchrone et semi-synchrone, sauf si $1 \leq i \leq j \leq 2$ ou si $i = j = 3$. Dans le premier de ces deux derniers cas, k doit être égal à $i \times j$; dans le second, k doit être au moins égal à 5. Nos résultats sont constructifs puisque nous proposons des algorithmes dans le modèle asynchrone utilisant le minimum de robots pour chacun des cas [§].

2 Bornes inférieures

Nous nous plaçons tout d'abord dans le modèle semi-synchrone. Par définition, toute impossibilité constatée dans ce modèle est également valide dans le modèle asynchrone.

Dans la suite, lorsque plusieurs robots sont localisés sur un même nœud de la grille, nous disons qu'ils forment une *tour*. Dans [2], les auteurs montrent que dans un système semi-synchrone où le nombre de nœuds et supérieur au nombre de robots, alors toute configuration finale doit pouvoir se distinguer de la configuration initiale (sans tour). D'où :

Remarque 1 *Un algorithme (déterministe ou probabiliste) d'exploration finie fonctionnant sur une grille $i \times j$ et utilisant k robots termine dans une configuration sans tour si et seulement si $k = i \times j$.*

En d'autres termes, le seul algorithme d'exploration fonctionnant sans construire de tour est l'algorithme trivial où il y a autant de robots que de nœuds. La remarque 1 nous permet également d'affirmer qu'il faut au moins 2 robots (afin de pouvoir former une tour) pour explorer toute grille constituée d'au moins 3 nœuds. Ensuite, en supposant que 2 robots soient suffisants, nous pouvons construire une exécution séquentielle possible e constituée d'un préfixe sans tour et d'une configuration terminale contenant une tour telle que e termine sans que la grille ait été entièrement visitée. D'où :

Théorème 1 *Il n'existe pas d'algorithme (déterministe ou probabiliste) d'exploration finie fonctionnant sur une grille $i \times j$ telle que $i \times j \geq 3$ qui utilise moins de 3 robots.*

La grille 2×2 étant aussi un anneau de 4 nœuds, le résultat de [2] nous permet d'affirmer que 4 robots sont nécessaires (et suffisants) pour visiter une telle grille. Ainsi :

Théorème 2 *L'exploration finie (déterministe ou probabiliste) d'une grille $i \times j$ telle que $1 \leq i \leq j \leq 2$ est possible si et seulement si $k = i \times j$.*

Les théorèmes 1 et 2 nous donnent la condition nécessaire de notre résultat global à l'exception de la grille 3×3. En supposant $k = 3$ robots, nous avons montré qu'il existe toujours des exécutions où au plus 3 nœuds sont visités avant de créer une tour d'au moins deux robots. Nous avons ensuite utilisé un model-checker qui nous a permis de générer de manière exhaustive toutes les classes de configurations atteignables par tout algorithme en partant de n'importe quelle configuration contenant une tour d'au moins deux robots. Nous avons ainsi montré qu'aucune des classes générées ne permettait de terminer l'exploration sans oublier de nœuds [¶]. En supposant $k = 4$ robots, il est facile de montrer que l'exploration peut ne pas terminer. En effet, en partant d'une configuration où les 4 robots sont initialement situés sur les nœuds médians des 4 bords, quel que soit l'algorithme, les choix de l'adversaire peuvent toujours mener à une configuration où soit chaque robot est dans un coin distinct, soit les robots forment tous une tour sur le nœud central de la grille. Dans ces deux cas, les choix de l'adversaire peuvent ensuite forcer les déplacements des quatre robots pour revenir à la configuration initiale. Les robots étant amnésiques, l'exécution ne termine alors jamais.

§. La version détaillée de cet article est disponible sur `http://arxiv.org/abs/1105.2461`.

¶. `http://www-verimag.imag.fr/~raymond/misc/robots/`

3 Exploration avec trois robots

Nous considérons maintenant le modèle asynchrone et nous proposons deux algorithmes (déterministes) d'exploration finie de grilles $i \times j$ n'utilisant que 3 robots. Le premier fonctionne sur toute grille $i \times j$ où $j > 3$. Le cas particulier de la grille 2×3 est présenté à la fin de cette section.

Algorithme général. L'algorithme fonctionne selon les 3 phases suivantes :

`Mise-en-place` : De loin la plus ardue, cette phase fait l'objet d'explications un peu plus détaillées ci-dessous. Partant d'une configuration quelconque sans tour, elle vise à mettre les 3 robots dans un des coins d'un des plus longs bords de la grille comme présenté dans la figure 1.

`Orientation` : Cette phase succède à la `Mise-en-place`. Cette phase est constituée d'un seul mouvement : le robot situé dans coin de la grille se déplace sur le nœud voisin contenant un robot et construit ainsi une tour. La figure 2 présente un exemple de configuration obtenue. Grâce à cette tour nous obtenons un système de coordonnées implicite, comme décrit dans la figure.

`Exploration` : La phase d'exploration utilise le système de coordonnées défini lors de la phase précédente pour explorer toute la grille sauf les nœuds $(0,0)$, $(0,1)$ et $(0,2)$ dont on sait qu'ils sont déjà visités. Durant cette phase, le robot initialement en $(0,2)$ se déplace comme décrit en figure 3. Il se déplace toujours vers le nœud (voisin) successeur de sa position courante suivant l'ordre :
$(x,y) \preceq (x',y') \equiv y < y' \vee [y = y' \wedge (x = x' \vee y \bmod 2 = 0 \wedge x < x' \vee y \bmod 2 = 1 \wedge x > x')]$. L'exploration se termine lorsqu'il a atteint le nœud maximum dans cet ordre.

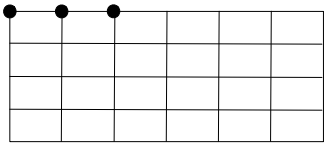

FIGURE 1: `Mise-en-place`

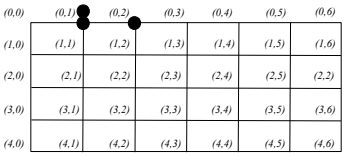

FIGURE 2: `Orientation`

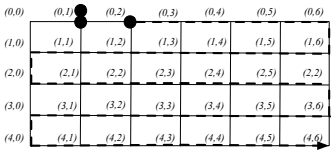

FIGURE 3: `Exploration`

L'espace alloué ne nous permet pas de décrire chaque phase en détail. Les phases d'orientation et d'exploration ne réclamant guère plus d'explication, nous nous concentrons donc sur les grandes lignes de la stratégie mise en œuvre lors de la phase `Mise-en-place`. L'algorithme réalise cette phase sans créer de tour, ce qui assure que cette phase est distincte des deux autres. Pour cela, il distingue trois classes de configurations.

La première de ces trois classes (appelons-la L) regroupe toutes les configurations où un seul robot est localisé dans un coin de la grille. Appelons ce robot r_1. Dans ce cas, le but est d'amener les deux autres robots (r_2 et r_3) sur l'un des plus longs côtés adjacents à r_1 – remarquons qu'il peut y en avoir 2 dans le cas d'une grille carré ($i = j$). Les déplacements de r_2 et r_3 sont ordonnés en fonction de la distance de Manhattan qui les séparent de r_1. Les deux robots rejoignent le nœud libre le plus proche sur le côté ciblé *via* le plus court chemin, avec priorité au robot le plus proche de r_1, s'il y en a un. Une fois que les robots sont tous sur le côté ciblé, ils se regroupent de manière à former la configuration finale attendue.

Bien sûr, cette stratégie peut être mise à mal par des situations particulières. Si r_2 et/ou r_3 sont localisés sur le plus petit côté adjacent à r_1, ils doivent au préalable s'écarter du bord d'un saut sur le côté, le cas échéant en évitant tout nœud déjà occupé.

Toujours dans la classe L, quelques cas de symétries peuvent aussi se produire, en particulier lorsque la grille est carrée. La situation la plus caractéristique est lorsque r_2 et r_3 sont chacun sur les bords orthogonaux se croisant là où est r_1. Si la distance les séparant de r_1 est supérieure à 1 (ils ne sont pas sur les nœuds adjacents à celui de r_1), c'est alors r_1 qui brise la symétrie en s'écartant du coin (le choix de la direction est fait par l'adversaire). Si r_2 et r_3 sont voisins de r_1, alors r_2 et r_3 ont pour consigne de s'écarter de r_1 en restant sur leur bord. Si les deux agissent de manière synchrone, alors on retrouve le cas précédent (et c'est r_1 qui finalement brise la symétrie), sinon la symétrie est directement brisée (soit r_2 soit r_3 n'est plus voisin de r_1).

La seconde classe de configurations, la classe C, regroupe les configurations où plusieurs robots sont situés sur des coins. Lorsqu'il y en a exactement deux, c'est au troisième robot de briser la symétrie si

nécessaire. S'il y en a 3, c'est le ≪ robot du milieu ≫ (situé à l'intersection des deux côtés où sont positionnés les 3 robots) qui brise la symétrie.

Enfin, il ne reste plus qu'une seule classe de configurations, la classe N où aucun des coins n'est occupé. Là encore, c'est une stratégie basée sur un ordonnancement des 3 robots qui est mise en œuvre, elle-même fondée sur les distances des robots par rapport aux coins. Les cas de symétries sont une nouvelle fois brisés en élisant un des 3 robots. En particulier, si deux robots sont équidistants d'un même coin, c'est le troisième robot qui est élu pour briser la symétrie. Lorsque les 3 robots sont équidistants de 3 coins distincts, c'est le robot le plus proche du coin situé entre les deux autres coins qui est élu.

Algorithme pour la grille 2×3. La première étape de cet algorithme consiste aussi à aligner les trois robots. S'ils ne sont pas alignés dès le départ, alors deux sont sur le même ≪ grand ≫ bord et le troisième se déplace alors pour compléter l'alignement. Ensuite, le robot du milieu bouge vers un de ces voisins pour créer une tour (le choix de voisin est opéré par l'adversaire). Enfin, le robot resté seul explore l'autre ≪ grand ≫ bord et l'algorithme termine.

Les deux stratégies décrites dans cette section nous permettent d'établir le théorème suivant :

Théorème 3 *Les deux algorithmes décrits dans la section 3 résolvent de manière déterministe le problème de l'exploration finie avec seulement* 3 *robots pour toute grille $i \times j$ telle que $j \geq 3$ et $j = 3 \Rightarrow i \neq 3$.*

4 Exploration de la grille 3×3 avec cinq robots

L'algorithme (toujours dans le modèle asynchrone) fonctionne en 2 phases : la phase de `Préparation` qui a pour but d'amener les 5 robots dans l'une des 3 configurations de départ de la phase d'`Exploration`. Ces 3 configurations, ainsi que la phase d'exploration sont décrites par la figure 4. La phase de préparation utilise le fait qu'avec 5 robots, il est souvent possible de désigner une ligne, dite *ligne directrice*, contenant le maximum de robots et d'utiliser cette ligne pour orienter la grille 3×3 et amener les 5 robots à former l'une des 3 configurations attendues. Nous utilisons le terme ≪ *souvent* ≫ car il y a des configurations pour lesquelles aucune ligne directrice ne peut être déterminée. C'est par exemple le cas (i) lorsque les 5 robots occupent 2 côtés adjacents ou bien (ii) lorsque 4 robots sont sur les 4 coins, le cinquième étant au centre. Mais il est toujours possible de casser la symétrie et ainsi d'aller vers l'une des 3 configurations recherchées. Par exemple, le cas (i) est résolu en éloignant les 2 robots situés sur les 2 coins vers les 2 côtés opposés. Le cas (ii) est quand à lui résolu en déplaçant le robot situé au centre vers l'un des bords.

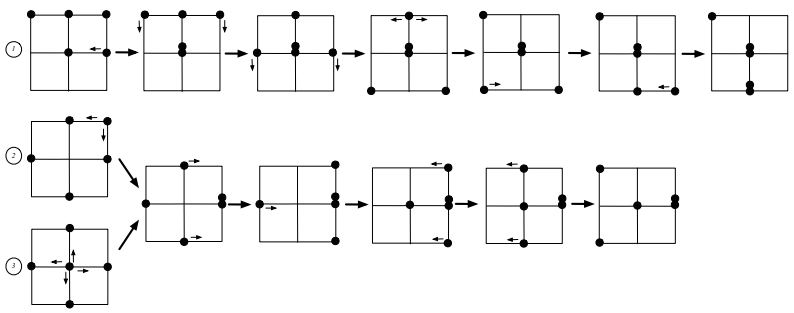

FIGURE 4: Phase d'exploration en partant d'une configuration 1, 2 ou 3.

Théorème 4 *L'algorithme décrit dans la section 4 résout de manière déterministe le problème de l'exploration finie avec seulement* 5 *robots sur la grille 3×3.*

Références

[1] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating : Ring exploration by asynchronous oblivious robots. *Algorithmica*, pages 1–22, 2012.

[2] Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In *Proceedings of the International Colloquium on Structural Information and Communication Complexity (SIROCCO 2009)*, volume 5869 of *Lecture Notes in Computer Science*, pages 195–208. Springer-Verlag Berlin Heidelberg, 2009.

[3] Anissa Lamani, Maria Potop-Butucaru, and Sébastien Tixeuil. Optimal deterministic ring exploration with oblivious asynchronous robots. In *Proceedings of the International Colloquium on Structural Information and Communication Complexity (SIROCCO 2010)*, volume 6058 of *Lecture Notes in Computer Science*, pages 183–196. Springer Berlin / Heidelberg, 2010.